SÜTRON
visualization | control | communication

User Manual

Scripting with TSwin .net

Part Number:	80860.703
Version:	1
Date:	05.02.2007
Valid for:	TSwin .net 4.1 + SP3

visualization | control | communication

Version	Date	Modifications
1	05.02.2007	First Edition

This manual, including all illustrations contained herein, is copyright protected. Use of this manual by any third party in departure from the copyright provision is forbidden. No part of this manual may be reproduced, translated or electronically or photographically archived or altered without the express written consent from Sütron electronic GmbH. Violations shall be cause for damage liability.
Sütron electronic reserves the right to make any changes that contribute to technical improvement.

Overall Table of Contents

1 Important Notes 1-1
1.1 Symbols 1-1
1.2 Safety Notes 1-1
1.3 Intended Use 1-1
1.4 Target Group 1-1

2 Scripting with TSwin .net 2-1
2.1 Starting Scripts 2-3
2.2 Variables in Scripts 2-5
2.3 Script Editor 2-6
2.4 Namespaces 2-7
2.5 Classes 2-7
2.6 Methods 2-8
2.6.1 Methods for ControllerVar Class 2-8
2.6.2 Methods for FileAccess Class 2-15
2.6.3 Methods for FlashFileSystem Class 2-31
2.6.4 Methods for PLCSymbolicAddr Class 2-35
2.6.5 Methods for ScriptVar Class 2-50
2.6.6 Methods for ScriptVarString Class 2-59
2.6.7 Methods for SysVar Class 2-66
2.7 Data Types for Scripts 2-69
2.8 Keywords 2-69
2.9 Operators 2-70
2.10 Comparison Operators 2-70
2.11 Control Structures 2-71
2.12 Local Variables 2-72
2.13 Global Variables 2-72

A Index A-1

1 Important Notes

1.1 Symbols

The symbols in this manual are used to draw your attention on notes and dangers.

Danger
This symbol is used to refer to instructions which, if ignored or not carefully followed could result in personal injury.

Note
This symbol indicates application tips or supplementary notes.

Reference to source of information
This symbol refers to detailed sources of information on the current topic.

1.2 Safety Notes

- Read this manual carefully before using the software. Keep this manual in a place where it is always accessible to all users.
- The user manual, in particular the safety notes, must be observed by all personnel working with the software and the programmed device.
- Observe the accident prevention rules and regulations that apply to the operating site.
- Installation and operation must only be carried out by qualified and trained personnel.

1.3 Intended Use

- The software has to be used for programming operating devices exclusively. Every other use is not permitted.

1.4 Target Group

All configuration and programming work in connection with the automation system must be performed by trained personnel only (e.g. qualified electricians, electrical engineers).

The configuration and programming personnel must be familiar with the safety concepts of automation technology.

2 Scripting with TSwin .net

Scripts are structured programs that are processed procedurally by an interpreter at runtime. You can use scripts to implement your own ideas, thus allowing you to derive greater benefit from your applications.

Since scripts are procedurally processed, the processing of an individual script is completed once the last script line has been processed. However, you can use control structures to create loops, whose conditions may not ever be fulfilled, within a script. Although the operating device is not blocked as a result, it may display incorrect variable values.

You can perform the following actions in scripts:

– influence the values of global, local, script and controller variables (reading and writing)
– create, open, read, write and close files,
– use control structures to read conditions.

To create scripts you start the script editor within the programming software. This editor supports you writing structured scripts by

– suggesting automatic completions,
– putting tabulators automatically,
– displaying script items in different colors and
– marking errors.

If you see an error in a script you can navigate the mouse cursor over the marked area to get further informations.

The content of a script is limited to items which are permitted within the given namespace. Due to this you achieve only scripts which are able to run on the given hardware platform.

Scripts have a specific structure. The use of tab stops provides you with a better overview of the content.

Structure of a script:

```
1  public void Execute ()
2  {
3      int result = f1(0);
4  }
5  private int f1 (int y)
6  {
7      int x = 0;
8      int z = 0;
9      while (x<7)
10     {
11         z = y + x;
12         x++;
13     }
14     return z;
15 }
```

Figure 2-1 Sample script

1 Declaration of function with parameter transfer ()

2 Opening parenthesis for the following script content

3 Assignment of the results of f1 into the variable "result". The f1 method is transferred to the parameter (0).

4 Closing parenthesis for the script content

5 Declaration of the f1 method and transfer of the parameter (0) into the variable (int y). The "int" declaration causes the method to return an integer value.

6 Opening parenthesis for the contents of the f1 method

7 Declaration of the x variable with value assignment (0)

8 Declaration of the z variable with value assignment (0)

9 Start the control structure with the query parameter (x<7)

10 Opening parenthesis for the contents of the control structure

11 Assignment of the result of adding the y and x variables into the z variable

12 Operation (increment operation) executed at the x variable

13 Closing parenthesis for the contents of the control structure

14 Return of the value of z to the calling method

15 Closing parenthesis for the contents of the f1 method

Declaration:

Each script and each function within a script must begin with the declaration. The declaration specifies the range of validity in which the script or the function is used, whether parameters are transferred and the name of the script.

Table 2-1 Ranges of validity

Syntax	Function
public	Script that is started by an event or cyclical
private	Function that is called within a script

Parameters are either transferred to the function or returned by the function. The return of a parameter is initiated with the "return" keyword.

Table 2-2 Parameter transfers

Syntax	Function
public void	This function does not return any parameter
public bool public int public uint public double public string	A parameter with the specified data type of this function is returned
private void f1 (bool y) private void f1 (int y) private void f1 (uint y) private void f1 (double y) private void f1 (string y)	A parameter of the specified data type is transferred to this function and written to the y variable

You can choose any name for a function but you must pay attention to upper and lower-case spelling. However, a script that is called by an event or cyclical must have the name "Execute"! (see line 1 of the sample script)

2.1 Starting Scripts

Scripts are started by an event or as planned task. Those events are

– starting a screen,
– leaving an editor screen.

Scripts cannot start other scripts.

Defining events:

Starting a screen

To start a script when a screen should be displayed select a script (e.g. = GetString PLC) in the **Screen script** field of the screen parameters dialog.

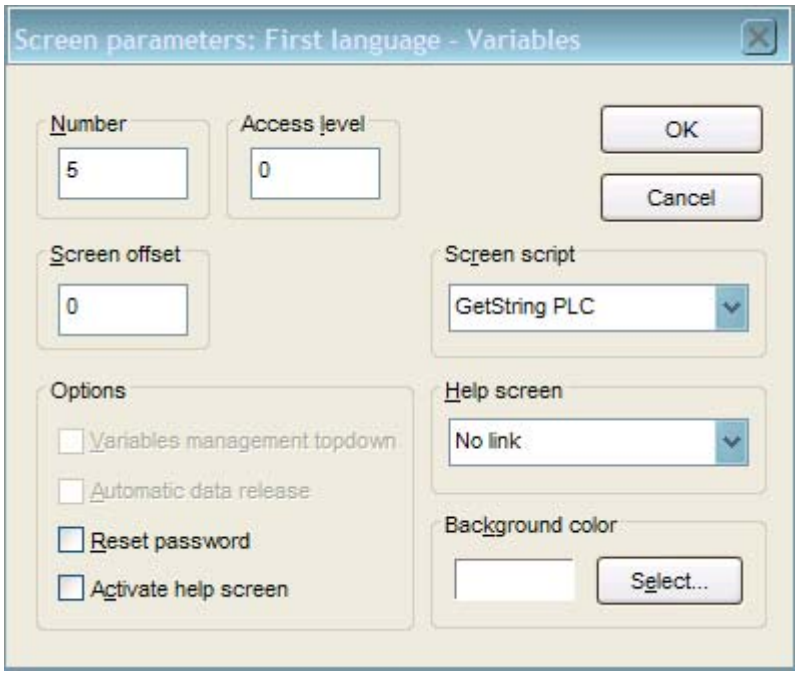

Figure 2-2 Screen parameters, screen script

Leaving an editor screen

If you will start a script after you have changed the value of variable (you leave the editor screen) select a script (e.g. = SetUInt PLC) in the **Post editing script** field of the representation type dialog (e.g. = Decimal number).

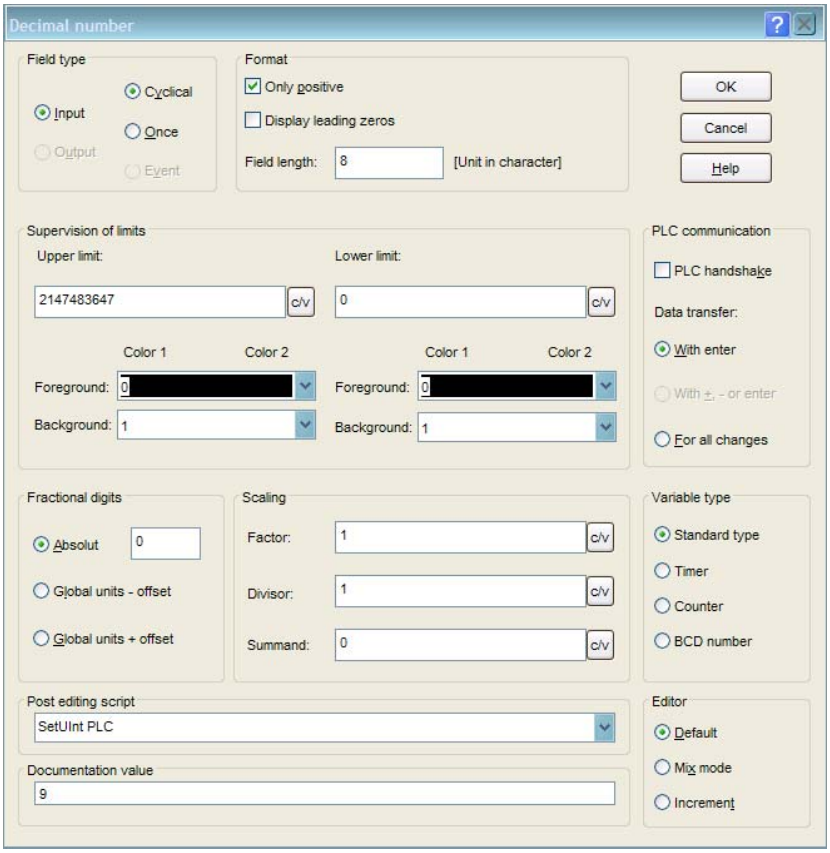

Figure 2-3 Representation type decimal, post editing script

Define planned tasks:

You define planned tasks to start scripts in a table.

Figure 2-4 Planed tasks

A task can be cyclical (see figure) or start after booting the operating device. You can define a time span for cyclical tasks till the task starts again.

2.2 Variables in Scripts

Within scripts, you can use variables, that are classified as follows.

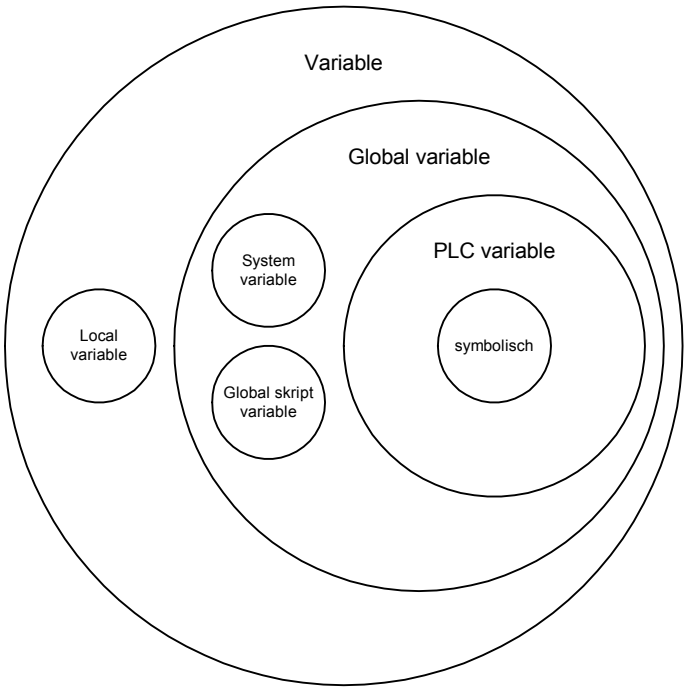

Figure 2-5 Classification of variables

Depending on the variable type, you must declare the variable at another position. Locations for the declaration include the script itself or the script variable list. Only system variables must not be declared, as their data types are known to the system.

Table 2-3 Variables for script processing

Type	Declaration location
Local script variable	In the script
Global script variable (retentive)	Variable list for script variables
System Variable	None (implicit)
Controller variable	Variable list for controller variables

Since you cannot assign for all parameters of methods global script variables, it is necessary to write its values to local script variables and to use the local script variables as parameters.

Example:

```
/* Sütron electronic GmbH
 * Version 1, 26.07.2006
 *
 * Function:
 * Visualize a direct controller address.
 * Read-in the string of a direct PLC address
 * into a global script variable. The string length
 * is given from a global skript variable.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode       UINT        1
```

```
 *  counter     UINT        1
 *  stringin    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  mystring    Byte
 */

public void Execute()
{
    // Declaration of the local variables
    uint foo = 0u;
    uint bar = 0u;

    // Write number of characters to read from global to lokal variable
    bar = Firmware.ScriptVar.GetUInt(Project.ScriptVar.counter);

    // Read the string
    foo = Firmware.ControllerVar.GetString(Project.ScriptVar.rcode,
                                           Project.ScriptVar.stringin,
                                           Project.ControllerVar.mystring,
                                           bar);
}
```

2.3 Script Editor

Create your scripts either with the internal script editor of the programming system or any external editor.

To overtake scripts edited by means of an external editor import its contents via the clipboard into the internal script editor.

The internal script editor has the advantage to offer automatically completion of certain text. This will reduce your work and also produces less typing errors. The following automatically completions are offered:

Classes names after namespaces:

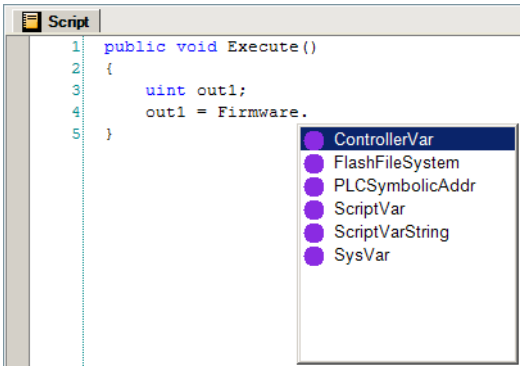

Figure 2-6 Automatically offer for classes names

Methods names after classes names:

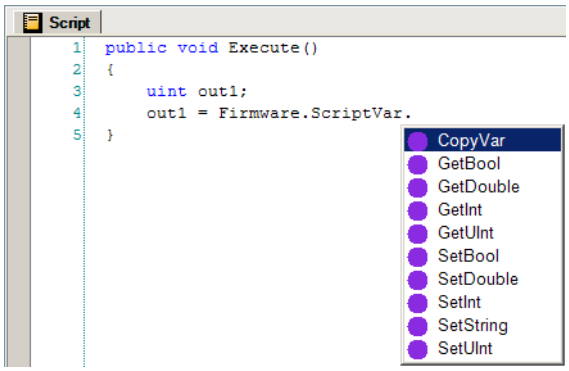

Figure 2-7 Automatically offer for methods names

Variables names after methods names:

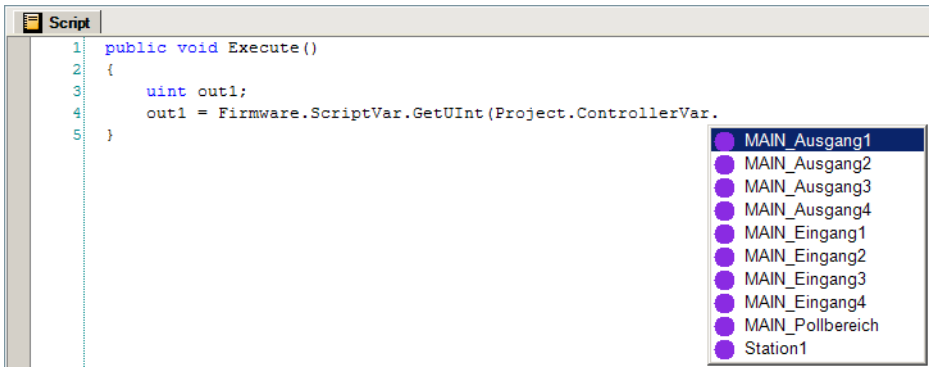

Figure 2-8 Automatically offer for variables names

2.4 Namespaces

You can use the following namespaces within the script:

– Firmware - for script elements that access elements, directly or indirectly, that are provided by the firmware.
– Project - for all other script elements

2.5 Classes

You can use the following classes within the namespaces:

– ControllerVar - provides access to native controller addresses
– FileAccess - provides access to files
– FlashFileSystem - provides access to the flash file system
– PLCSymbolicAddr - provides access to symbolic controller addresses (only for BRC-Symbolic)
– ScriptVar - provides access to script variables, local and global
– ScriptVarString - provides access to string-type script variables
– SysVar (firmware) - provides access to system variables

2.6 Methods

Methods are like program functions. To receive the value of a system variable or a string in a script, you must call a method that returns the requested value. Additional information, for example, whether the method was successfully applied, can also be returned along with the requested value. In this case, the return value could either be true(1) or false (0).

Methods depend on the namespace and the class where they are valid. Due to this they must always be called in the same manner:

Namespace.Class.Methodename(Parameter1, Parameter2);

The parameters that are submitted to the method can for their part be bound to a namespace again which you must indicate in any case.

Example of the method SetUInt of the class ScriptVar:

```
Firmware.ScriptVar.SetUInt(Project.ScriptVar.myvariable, 22u);
```

Depending on the class you can call different methods.

All of the example scripts in the following sections are tested using different controllers. You can mark the script code in the online help and copy it to a new script by means of the clipboard.

2.6.1 Methods for ControllerVar Class

With the methods of this class, you access the controller variables from the script. The controller variables must already be present in the variable list for controller variables.

The variable names must be conform to the following restrictions.

– Only the characters a to z, A to Z, 0 to 9 and _ are permitted.
– A variable name must not begin with a number (e.g. 2Stations).
– Each variable name must be unambiguous.

If a variable name fails to meet the restrictions, the programming software automatically changes it.

Examples:

Table 2-4 Conversion of variables names

Original Name	Converted Name in the Script	Comment
Machine	Project.ControllerVar.Machine	Unchanged
!Caution!	Project.ControllerVar._Caution_	Symbol is changed
Overvoltage	Project.ControllerVar.Overvoltage	Unchanged
Overvoltage	Project.Controller-Var.Overvoltage_2	Already defined, made unambiguous by _2
dwordarray[20]	Project.Controller-Var.dwordarray_20_	Symbols are changed
2Stations	Project.ControllerVar._Stations	Number at the beginning is changed

In the event of an error an error code is written to the return variable.

– the two high bytes contain the code of the COMMUNICATION ERROR,
– the two low bytes contain the subcode of the COMMUNICATION ERROR.

The code/subcode depends on the communication protocol that is currently used.

Example of how to divide the highest significant bytes from the lower significant bytes:

Divides the returncode of a method into higher significant bytes (code) and lower significant bytes (subcode).

```
/* Sütron electronic GmbH
 * Version 1, 17.08.2006
 *
 * Function:
 * Separates the returncode of a methode
 * into hi-word and low-word
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 */

public void Execute()
{
    // Declaratation of local script variables
    uint returncode = 0u;
    uint Code = 0u;
    uint Subcode = 0u;

    // Write the returnvariable value to a local script variable
    returncode = Firmware.ScriptVar.GetUInt(Project.ScriptVar.rcode1);

    // Separate the returncode into hi-word and low-word
    Code    = returncode >> 16;
    Subcode = returncode & 0x0000FFFFu;
}
```

GetBool(return variable, controller variable)

Reads the value of a controller variable from the data type bool.

Method	Return type
GetBool	bool
Return value = value of the variable	

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint

GetInt(return variable, controller variable)

Reads the value of a controller variable as a signed integer with 32 bits.

Method	Return type
GetInt	int
Return value = value of the variable	

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint

GetUInt(return variable, controller variable)

Reads the value of a controller variable as an unsigned integer with 32 bits.

Method	Return type
GetUInt	uint
Return value = value of the variable	

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint

GetDouble(return variable, controller variable)

Reads the value of a controller variable as an IEEE floating point number with 8 bytes.

Method	Return type
GetDouble	double
Return value = value of the variable	

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint

GetString(return variable, target script variable, controller variable, counter)

Copies a string from a controller variable to a target script variable. The counter parameter contains the number of characters to read.

Method	Return type
GetString	uint
Return value = length of the string (number of characters)	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
Controller variable	uint
Counter	uint

Example of ControllerVar.GetString:

Writes the string of a PLC address to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 25.07.2006
 *
 * Function:
 * Visualize a direct controller address.
 * Read-in the string (20 characters)of a
 * direct PLC address into a global script
 * variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode       UINT        1
 *  mystring    STRING      20
 *  length      UINT        1
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  plcstring   Byte
 */

public void Execute()
{
```

```
    // Declaration of the local variables
    uint foo = 0u;

    // Read the string
    foo = Firmware.ControllerVar.GetString(Project.ScriptVar.rcode1,
                                           Project.ScriptVar.mystring,
                                           Project.ControllerVar.plcstring,
                                           20u);

    // Write length of string to global skript variable
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.length, foo);
}
```

Depending on the controller and controller communication setting, the string is coded as a 8 bit ASCII or 16 bit UNICODE string!

SetBool(return variable, controller variable, value)

Writes a boolean value into a controller variable.

Method	Return type
SetBool	void

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint
Value	bool

SetInt(return variable, controller variable, value)

Writes a signed integer with 32 bits into the controller variable.

Method	Return type
SetInt	void

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint
Value	int

SetUInt(return variable, controller variable, value)

Writes an unsigned integer with 32 bits into the controller variable.

Method	Return type
SetUInt	void

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint
Value	uint

Example of ControllerVar.SetUInt:

Writes the value of a global script variable to a PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 25.07.2006
 *
 * Function:
 * Write a value to a direct controller address.
 * Read-in the value of a global script variable
 * into a local script variable and write the
 * value to a direct PLC address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  counter     UINT        1
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  mybyte      Byte
 */

public void Execute()
{

    // Declaration of the local variable
    uint foo = 0u;

    // Read the value into local variable
    foo = Firmware.ScriptVar.GetUInt(Project.ScriptVar.counter);

    // Write the value
    Firmware.ControllerVar.SetUInt(Project.ScriptVar.rcode,
                                   Project.ControllerVar.mybyte,
                                   foo);
}
```

SetDouble(return variable, controller variable, value)

Writes an IEEE floating point number with 8 bytes to the controller variable.

Method	Return type
SetInt	void

Parameter	Type
Return variable (global script variable)	uint
Controller variable	uint
Value	double

SetString(return variable, target controller variable, source script variable, counter)

Copies the string from the script variable to the controller variable. The counter parameter contains the number of characters to be written.

The return variable contains the length of the string that is read (number of characters).

Method	Return type
SetString	uint

Parameter	Type
Return variable (global script variable)	uint
Target controller variable	uint
Source script variable (global script variable)	uint
Counter	uint

Example of ControllerVar.SetString:

Writes the string of a global script variable to a PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 26.07.2006
 *
 * Function:
 * Write a string (20 characters) from a global script variable
 * to a direct controller address.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode       UINT        1
 *  mystring    STRING      20
 *  length      UINT        1
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  plcstring    Byte
 */

public void Execute()
{
    // Declaration of the local variables
    uint foo = 0u;

    // Read the string
    foo = Firmware.ControllerVar.SetString(Project.ScriptVar.rcode1,
                                           Project.ControllerVar.plcstring,
                                           Project.ScriptVar.mystring,
                                           20u);

    // Write length of string to global skript variable
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.length, foo);
}
```

Depending on the controller and controller communication setting, the string is coded as a 8 bit ASCII or 16 bit UNICODE string!

2.6.2 Methods for FileAccess Class

With the methods of this class you access files from the script.
You can

– open files.
– close files.
– write data of different types to files.
– read data of different types from files.

In the event of an error an error code that is analogue to the Win API function GetLastError() is written to the return variable.

The access on files is limited to these possibilities:

– CompactFlash card: **\StorageCard**
– USB stick (XScale with Win CE 4.2): **\Hard Disk**
– USB stick (ARM9 and XScale with Win CE 5.0): **\HardDisk**
– internal flash memory: **\FlashDrv**
– internal temporary directory: **\Temp**
– net drive: **//mypath/**

FileOpen(return variable, script variable, mode)

Opens a file with the name stored in the script variable. With mode you declare the type of access, combinations of access modes are possible (ra, rb, rt, wb, wt).

The return value contains the file handle for the opened file.

Method	Return type
FileOpen	uint

Parameter	Type
Return variable (global script variable)	uint
Script variable (global script variable)	uint
Mode r = read w = write a = append t = text b = binary c = enable commit flag n = reset commit flag	string

Example of FileOpen:

Opens a file for binary appending. The filename is given in the script. If you like to enter the filename on the screen you must create a screen containing an alphanumerical input variable (with a reference on the script variable named myfile) with an accordingly length and you must delete the script line below **//Setting the filename**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file for appending binary.
 * The filename is assigned in the skript.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myfile      STRING      25
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"/myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "ab");
}
```

FileClose(return variable, file handle)

Closes the file with the file handle of the **FileOpen** method.

Method	Return type
FileClose	uint

Parameter	Type
Return variable (global script variable)	uint
File handle	uint

Example of FileClose:

Closes the file whose file handle was created with FileOpen before.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Open a file for appending binary and close it.
 * The filename is assigned in the skript.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myfile      STRING      25
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"/myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "ab");
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode1, filehandle);
}
```

FileReadBool(return variable, target script variable, file handle)

Reads a boolean value from the file with the file handle of the **FileOpen** method. The target script variable contains the data.

Method	Return type
FileReadBool	uint
Return value >0 = number of read data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

Example of FileReadBool:

Opens a file, reads a boolean value and closes the file The value is written to the global script variable **mybool**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and reads a boolean value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mybool      BOOL        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "rb");

    // Read the boolean value
    returnvalue = Firmware.FileAccess.FileReadBool(Project.ScriptVar.rcode2,
                                                   Project.ScriptVar.mybool,
                                                   filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileReadInt(return variable, target script variable, file handle)

Reads an integer value from the file with the file handle of the **FileOpen** method. The target script variable contains the value.

Method	Return type
FileReadInt	uint
Return value >0 = number of read data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

Example of FileReadInt:

Opens a file, reads an integer value and closes the file. The value is written to the global script variable **myint**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and reads an integer value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  myint       INT         1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "rb");

    // Read the integer value
    returnvalue = Firmware.FileAccess.FileReadInt(Project.ScriptVar.rcode2,
                                                  Project.ScriptVar.myint,
                                                  filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileReadUint(return variable, target script variable, file handle)

Reads an unsigned integer from the file with the file handle of the **FileOpen** method. The target script variable contains the data.

Method	Return type
FileReadUint	uint
Return value >0 = number of read data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

Example of FileReadUInt:

Opens a file, reads an unsigned integer value and closes the file. The value is written to the global script variable **myuint**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and reads an unsigned integer value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  myuint      UINT        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                          Project.ScriptVar.myfile, "rb");

    // Read the unsigned integer value
    returnvalue = Firmware.FileAccess.FileReadUint(Project.ScriptVar.rcode2,
                                                   Project.ScriptVar.myuint,
                                                   filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileReadDouble(return variable, target script variable, file handle)

Reads an value of the type double from the file with the file handle of the **FileOpen** method.

Method	Return type
FileReadDouble	uint
Return value >0 = number of read data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint

Example of FileReadDouble:

Opens a file, reads a double value and closes the file. The value is written to the global script variable **mydouble**.

```
/* Sütron electronic GmbH
 * Version 2, 20.10.2006
 *
 * Function:
 * Opens a file and reads a double value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mydouble    DOUBLE      1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "rb");

    // Read the double value
    returnvalue = Firmware.FileAccess.FileReadDouble(Project.ScriptVar.rcode2,
                                                     Project.ScriptVar.mydouble,
                                                     filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileReadString(return variable, target script variable, file handle, counter)

Reads a string from the file with the file handle of the **FileOpen** method. The counter parameter contains the number of characters (16 bit) to read without termination. Enter a "0" for the counter parameter to read a string with "\0" termination.

Method	Return type
FileReadString	uint
Return value >0 = Number of read characters (without termination) Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
File handle	uint
Counter	uint

Example of FileReadString:

Opens a file, reads a String of variable length (up to 32 characters) and closes the file. Each string of the file has to end with cr/lf. If you enter a line number, that is too low the error text **Eingabefehler** (input error) appears. The end of the file is indicated with **EndOfFile**.

To operate the script you will need tree buttons on a screen which contain the global script variables **myfile** and **linecounter** as cyclical input variables and stringin as a cyclical output variable. Please select the **alphanumeric** representation type for both variables **myfile** and **stringin**. For the **linecounter** variable select the **decimal** representation type. Assign the script that is printed below as the post editing script of the **linecounter** variable.

```
/* Sütron electronic GmbH
 * Version 1, 31.07.2006
 *
 * Function:
 * Read a String from a file and write it to a global script variable.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mystring    STRING      20
 *  linecounter UINT        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    bool eof = false;
    bool eol = false;
    uint linecount = 0u;
    uint linelength = 0u;

    // Initialize global script variables
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.rcode1, 0u);
```

```
Firmware.ScriptVar.SetUInt(Project.ScriptVar.rcode2, 0u);
Firmware.ScriptVar.SetUInt(Project.ScriptVar.rcode3, 0u);

// Open the file
filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                          Project.ScriptVar.myfile, "rb");

// Determine if line number is too low
if (Firmware.ScriptVar.GetUInt(Project.ScriptVar.linecounter) <= 0u)
{
    Firmware.ScriptVar.SetString(Project.ScriptVar.stringin, "Eingabefehler");
    linelength = 0u; // Clear length of last line
    return;
}
else
{
    // Loop to read string
    while (eof == false)
    {
        linecount++;
        if (Firmware.ScriptVarString.GetChar(Project.ScriptVar.onechar,0u) == 10d)
        {
            Firmware.ScriptVar.SetString(Project.ScriptVar.stringin, "");
            eol = false;
        }

        // Loop to read one line - until carriage return
        while (eol == false)
        {
            // Increment counter of line length
            linelength++;

            // Clear character variable for input
            Firmware.ScriptVar.SetString(Project.ScriptVar.onechar, "");

            // Read one character from file
            Firmware.FileAccess.FileReadString(Project.ScriptVar.rcode2,
                                               Project.ScriptVar.onechar,
                                               filehandle,1u);

            if (Firmware.ScriptVarString.GetChar(Project.ScriptVar.onechar,0u) == 0d)
            {
                Firmware.ScriptVar.SetString(Project.ScriptVar.stringin,"EndOfFile");
                eol = true; // End of loop
            }
            else
            {
            // Add the single character to the string
            Firmware.ScriptVarString.CopySubstring(Project.ScriptVar.stringin,
                                                   Project.ScriptVar.onechar,0u);
            }
            // Determine end of line (carriage return)
            if (Firmware.ScriptVarString.GetChar(Project.ScriptVar.onechar,0u) == 10d)
            {
                linelength = 0u;
                eol = true; // End of loop
            }
        }

        // Determine if line number is reached
        if (linecount == Firmware.ScriptVar.GetUInt(Project.ScriptVar.linecounter))
        {
            eof = true; // End of loop
        }
```

```
        }

        // Close the file
        Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
    }
}
```

FileReadOneByte(return variable, target script variable, file handle)

Reads a byte from the file with the file handle of the **FileOpen** method. The read value is stored in the lowest significant byte of the target script variable. Afterwards the read position is increased by one byte.

In the event of an error the value 0 is written to the return variable.

Method	Return type
FileReadOneByte	uint

Parameter	Type
Return variable	uint
Target script variable	uint
File handle	uint

Example of FileReadOneByte:

Opens a file, reads a byte value and closes the file. The value is written to the global script variable **mybyte**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and reads a byte value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mybyte      UINT        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "rb");

    // Read the byte value (uint)
    returnvalue = Firmware.FileAccess.FileReadOneByte(Project.ScriptVar.rcode2,
                                                      Project.ScriptVar.mybyte,
                                                      filehandle);
    // Close the file
```

```
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileWriteBool(return variable, source script variable, file handle)

Writes a boolean value to the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteBool	uint
Return value >0 = number of read data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

Example of FileWriteBool:

Opens a file, writes a boolean value and closes the file. The value must be in the global script variable **mybool**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and writes a boolean value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mybool      BOOL        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "wb");

    // Write the boolean value
    returnvalue = Firmware.FileAccess.FileWriteBool(Project.ScriptVar.rcode2,
                                                    Project.ScriptVar.mybool,
                                                    filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileWriteInt(return variable, source script variable, file handle)

Writes an integer value to the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteInt	uint
Return value = number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

Example of FileWriteInt:

Opens a file, writes an integer value and closes the file. The value must be in the global script variable **myint**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and writes an integer value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  myint       INT         1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "wb");

    // Write the integer value
    returnvalue = Firmware.FileAccess.FileWriteInt(Project.ScriptVar.rcode2,
                                                   Project.ScriptVar.myint,
                                                   filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileWriteUint(return variable, source script variable, file handle)

Writes an unsigned integer value to the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteUInt	uint
Return value = number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

Example of FileWriteUInt:

Opens a file, writes an unsigned integer value and closes the file. The value must be in the global script variable **myuint**.

```
/* Sütron electronic GmbH
 * Version 2, 25.10.2006
 *
 * Function:
 * Opens a file and writes an unsigned integer value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  myuint      UINT        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "wb");

    // Write the unsigned integer value
    returnvalue = Firmware.FileAccess.FileWriteUint(Project.ScriptVar.rcode2,
                                                    Project.ScriptVar.myuint,
                                                    filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileWriteDouble(return variable, source script variable, file handle)

Writes a value of the type 0x83 FLOAT64 to the file with the file handle of the **FileOpen** method. The source script variable contains the value to be written.

Method	Return type
FileWriteDouble	uint
Return value = number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint

Example of FileWriteDouble:

Opens a file, writes a double value and closes the file. The value must be in the global script variable **mydouble**.

```
/* Sütron electronic GmbH
 * Version 2, 20.10.2006
 *
 * Function:
 * Opens a file and writes a double value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mydouble    DOUBLE      1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "wb");

    // Write the boolean value
    returnvalue = Firmware.FileAccess.FileWriteDouble(Project.ScriptVar.rcode2,
                                                      Project.ScriptVar.mydouble,
                                                      filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

FileWriteString(return variable, source script variable, file handle, counter)

Writes a string to the file with the file handle of the **FileOpen** method. The counter parameter contains the number of characters (16 bit) to be written either with or without termination. Enter a "0" for the counter parameter to write a string with "\0" termination to the file.

Method	Return type
FileWriteString	uint
Return value = number of written data in byte	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
File handle	uint
Counter	uint

Example of FileWriteString:

Opens a file, writes a string of variable length (up to 20 characters) and closes the file. Each string contains a cr/lf at its end to enable easy reading at a later time.

To operate the script you will need two buttons on a screen which contain the global script variables **myfile** and **mystring** as cyclical input variables. Please select the **alphanumeric** representation type for both variables. Assign the script that is printed below as the post editing script of the **mystring** variable.

```
/* Sütron electronic GmbH
 * Version 1, 28.07.2006
 *
 * Function:
 * Write a String from a global script variable to a file.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode       UINT        1
 *  myfile      STRING      25
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;

    // Open the file
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode,
                                              Project.ScriptVar.myfile, "a");

    // Write the string to the file
    Firmware.FileAccess.FileWriteString(Project.ScriptVar.rcode,
                                        Project.ScriptVar.mystring,
                                        filehandle,
                                       Firmware.ScriptVarString.Length(Project.ScriptVar.mystring));

    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode, filehandle);

    // Clear the global script variable
```

```
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring, "");
}
```

FileWriteOneByte(return variable, source script variable, file handle)

Writes the lowest significant byte of the source script variable to a file with the file handle of the **FileOpen** method.

The return value indicates the number of written bytes.

Method	Return type
FileWriteOneByte	uint

Parameter	Type
Return variable	uint
Source script variable	uint
File handle	uint

Example of FileWriteOneByte:

Opens a file, writes a byte value and closes the file. The value must be in the global script variable **mybyte**.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Opens a file and writes a byte value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  rcode2      UINT        1
 *  rcode3      UINT        1
 *  myfile      STRING      25
 *  mybyte      UINT        1
 */

public void Execute()
{
    // Declaration of the local variable
    uint filehandle = 0u;
    uint returnvalue = 0u;

    // Setting the filename
    Firmware.ScriptVar.SetString(Project.ScriptVar.myfile,"myfilename.dat");

    // Open the file named directly in the script
    filehandle = Firmware.FileAccess.FileOpen(Project.ScriptVar.rcode1,
                                              Project.ScriptVar.myfile, "wb");

    // Write the byte value of the variable mybyte
    returnvalue = Firmware.FileAccess.FileWriteOneByte(Project.ScriptVar.rcode2,
                                                       Project.ScriptVar.mybyte,
                                                       filehandle);
    // Close the file
    Firmware.FileAccess.FileClose(Project.ScriptVar.rcode3, filehandle);
}
```

2.6.3 Methods for FlashFileSystem Class

With the methods of this class, you access the FlashFile system from the script. This allows you to store Drive parameters from the controller to the FlashFile system of the small operator terminal and to write them back to the controller.

To store Drive parameters from the controller onto the terminal, you need to call up the following functions in sequence:

1. ResetFFS
2. InitSaveDriveParam
3. SaveDriveParam
4. CloseBackupDriveParam

To transfer Drive parameters from the terminal onto the controller, you need to call up these functions in sequence:

1. InitRestoreDriveParam
2. ReadBytesToRestore
3. RestoreDriveParam

In the event of an error an error code is written to the return variable.

– the two high bytes contain the error code,
– the two low bytes contain the error subcode.

The error codes / subcodes are described for each method.

ResetFFS(return variable)

Locks the interface for the application and then erases the memory of the FlashFile system.

Method	Return type
ResetFFS	uint
Return value >0 = ok Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint

Table 2-5 Return variable for method ResetFFS

Error code	Error subcode	Description
255	3	Wrong flash
	6	Error erasing flash

InitSaveDriveParam(return variable, station address)

Initializes the backup of the Drive parameters which are stored in controller at the station address.

Method	Return type
InitSaveDriveParam	uint
Return value >0 = number of bytes to save Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint

Table 2-6 Return variable for method InitSaveDriveParam

Error code	Error subcode	Description
255	1	Wrong length
	2	File write error
	3	File open error
	4	File close error
0	0	No data to save

SaveDriveParam(return variable, station address, start element, number of elements)

Reads a block of elements (a maximum of 1400) - starting from the number of the start element - which are located at the station address of the controller.

If more than 1400 elements are to be read, it might be necessary to start this method several times in order to obtain the complete set of parameters.

Method	Return type
SaveDriveParam	uint
Return value >0 = operation ok Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint
Start element	uint
Number of elements	uint

Table 2-7 Return variable for method SaveDriveParam

Error code	Error subcode	Description
255	1	Wrong length
	2	File write error
	3	File open error
	4	File close error
	5	Insufficient memory

CloseBackupDriveParam(return variable, station address)

Terminates the transfer of Drive parameters stored at the station address of the controller and opens the interface for the application.

Method	Return type
CloseBackupDriveParam	uint
Return value 1 = operation ok Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint

Table 2-8 Return variable for method CloseBackupDriveParam

Error code	Error subcode	Description
255	3	File open error
	4	File close error

InitRestoreDriveParam(return variable, station address)

Initializes the transfer of Drive parameters to the station address in the controller.

Method	Return type
InitRestoreDriveParam	uint
Return value 1 = operation ok Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint

Table 2-9 Return variable for method InitRestoreDriveParam

Error code	Error subcode	Description
255	3	Wrong flash

ReadBytesToRestore(return variable)

Reads the number of bytes to be written to the controller.

Method	Return type
ReadBytesToRestore	uint
Return value >0 = number of data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint

Table 2-10 Return variable for method ReadBytesToRestore

Error code	Error subcode	Description
255	1	Wrong length
	2	File write error
	3	File open error
	4	File close error
0	0	No data to save

RestoreDriveParam(return variable, station address, start element, number of elements)

Reads a block of elements (a maximum of 1400) - starting from the number of the start element - which are located at the station address of the controller.

If more than 1400 elements are to be written, it might be necessary to start this method several times in order to transfer the complete set of parameters.

Method	Return type
RestoreDriveParam	uint
Return value >0 = number of data in byte Return value 0 = error	

Parameter	Type
Return variable (global script variable)	uint
Station address	uint
Start element	uint
Number of elements	uint

Table 2-11 Return variable for method RestoreDriveParam

Error code	Error subcode	Description
255	1	Wrong length
	2	File write error
	3	File open error
	4	File close error
	5	File seek error
	6	File length error

2.6.4 Methods for PLCSymbolicAddr Class

With the methods of this class, you access - from the script - individual controller variables using symbolic addresses.

Supported are single accesses to a controller variable and single accesses to a field of a controller variable.

As opposed to direct addressed controller variables a type check takes place during the read process of symbolic addresses. The test fails since the type "Word" is unsigned, to read from a word address with "GetInt".

You store the access address in a script variable (address script variable; e.g. 10,5,S-0-0001.5). You can then use this script variable to access the controller variable. If the action is successful, the return value contains a 0 (zero).In the event of an error, the error number of the communication server is transferred as a return value.

Using the Element number parameter, you specify whether you are accessing a single variable (0) or a field (0x80000000 + element number).

The return variable contains 0 (Zero) if the method ended successfully, but the error code of the error message from the communication server if the method failed. You gather the meaning of the error code from the documentation to the corresponding connection.

GetListLength(return variable, address script variable, station address)

Determines the length of a field variable.

For this purpose, a read access is performed to element 0 of the field with the type 0x09 C-STRING.

Method	Return type
GetListLength	uint

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint

Example of PLCSymbolicAddr.GetListLength:

Determines the length of a variable/field variable and writes it to a global script variable. If the determined length is 0 (zero) it is a single variable!

```
/* Sütron electronic GmbH
 * Version 1, 11.08.2006
 *
 * Function:
 * Determine the length of an array and write it
 * to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myvar       UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_array  ARRAY[0..15] OF BYTE
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_array");

   // Determine the length of plc_array
   foo = Firmware.PLCSymbolicAddr.GetListLength(Project.ScriptVar.rcode1,
                                                Project.ScriptVar.symbaddr, 1u);

   // Write value from local to global script variable
   Firmware.ScriptVar.SetUInt(Project.ScriptVar.myvar, foo);
}
```

GetBool(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x01 bool with a length of 1 byte.

Method	Return type
GetBool	bool

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.GetBool:

Writes the value of a symbolic PLC address of the boolean type to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 29.08.2006
 *
 * Function:
 * Read-in the value of a symbolic PLC address
 * into a local script variable and write the
 * value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mybool      BOOL        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_bool   BOOL
 */

public void Execute()
{
   // Declaration of local variable
   bool foo = false;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_bool");

   // Read-in the value from the symbolic address
   foo = Firmware.PLCSymbolicAddr.GetBool(Project.ScriptVar.rcode1,
                                          Project.ScriptVar.symbaddr, // Symbolic address
                                          1u,                         // Station number
                                          0u);                        // Element number

   // Write value from local to global script variable
   Firmware.ScriptVar.SetBool(Project.ScriptVar.mybool, foo);
}
```

GetInt(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x04 INT32 with a length of 4 bytes.

Method	Return type
GetInt	int

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.GetInt:

Writes the value of a symbolic PLC address of the integer type to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 08.08.2006
 *
 * Function:
 * Read-in the value of a symbolic PLC address
 * into a local script variable and write the
 * value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myint       INT         1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_int    INT
 */

public void Execute()
{
   // Declaration of local variable
   int foo = 0;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_int");

   // Read-in the value from the symbolic address
   foo = Firmware.PLCSymbolicAddr.GetInt(Project.ScriptVar.rcode1,
                                         Project.ScriptVar.symbaddr, // Symbolic address
                                         1u,                         // Station number
                                         0u);                        // Element number

   // Write value from local to global script variable
   Firmware.ScriptVar.SetInt(Project.ScriptVar.myint, foo);
}
```

GetUInt(return variable, address script variable, station address, element number)

Reads the value of a variable of the 0x07 UINT32 type with a length of 4 bytes.

Method	Return type
GetUInt	uint

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.GetUInt:

Writes the value of a symbolic PLC address to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 08.08.2006
 *
 * Function:
 * Read-in the value of a symbolic PLC address
 * into a local script variable and write the
 * value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myuint      UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_uint   UINT
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_uint");

   // Read-in the value from the symbolic address
   foo = Firmware.PLCSymbolicAddr.GetUInt(Project.ScriptVar.rcode1,
                                          Project.ScriptVar.symbaddr, // Symbolic address
                                          1u,                         // Station number
                                          0u);                        // Element number

   // Write value from local to global script variable
   Firmware.ScriptVar.SetUInt(Project.ScriptVar.myuint, foo);
}
```

Example of PLCSymbolicAddr.GetUInt for an array item:

Writes the value of an ARRAY item of a symbolic PLC address to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 14.08.2006
 *
 * Function:
 * Read-in the value of an element of an array
 * from a symbolic PLC address into a local script variable
 * and write the value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myvar       UINT        1
 *  counter     UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_array  ARRAY[0..15] OF BYTE
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;
   uint element = 0u;

   // Write element number to local script variable
   element = Firmware.ScriptVar.GetUInt(Project.ScriptVar.counter);

   // Define array element number
   element = 0x80000000 + element;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_array");

   // Read-in the value from the symbolic address
   foo = Firmware.PLCSymbolicAddr.GetUInt(Project.ScriptVar.rcode1,
                                  Project.ScriptVar.symbaddr, // Symbolic address
                                  1u,                         // Station number
                                  element);                   // Element number

   // Write value from local to global script variable
   Firmware.ScriptVar.SetUInt(Project.ScriptVar.myvar, foo);
}
```

GetDouble(return variable, address script variable, station address, element number)

Reads the value of a variable of the type 0x83 FLOAT64 with a length of 8 bytes.

Method	Return type
GetDouble	double

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.GetDouble:

Writes the value of a symbolic PLC address to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 30.08.2006
 *
 * Function:
 * Read-in the value of a symbolic PLC address
 * into a local script variable and write the
 * value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mydouble    UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_double REAL
 */

public void Execute()
{
   // Declaration of local variable
   double foo = 0d;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_double");

   // Read-in the value from the symbolic address
   foo = Firmware.PLCSymbolicAddr.GetDouble(Project.ScriptVar.rcode1,
                                            Project.ScriptVar.symbaddr, // Symbolic address
                                            1u,                         // Station number
                                            0u);                        // Element number

   // Write value from local to global script variable
   Firmware.ScriptVar.SetDouble(Project.ScriptVar.mydouble, foo);
}
```

GetString(return variable, target script variable, address script variable, station address, element number)

Reads the value of a variable of the 0x09 C-STRING type.

Method	Return type
GetString	uint
Return value = length of the string (number of characters)	

Parameter	Type
Return variable (global script variable)	uint
Target script variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.GetString:

Writes the string of symbolic PLC address to a global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 13.09.2006
 *
 * Function:
 * Read-in the string of a symbolic PLC address
 * into a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mystring    STRING      20
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_string STRING
 */

public void Execute()
{
    // Write symbolic address to global script variable
    Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_string");

    // Read-in the value from the symbolic address
    Firmware.PLCSymbolicAddr.GetString(Project.ScriptVar.rcode1,
                                       Project.ScriptVar.mystring, // Target variable
                                       Project.ScriptVar.symbaddr, // Symbolic address
                                       1u,                         // Station number
                                       0u);                        // Element number
}
```

SetBool(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the type 0x01 bool with a length of 1 byte.

Method	Return type
SetBool	void
Return value = TRUE	

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	bool
Station address	uint
Element number	uint

You have to return the values **false** or **true**.
To return these values to a global script variable (0 or 1) you have to proceed a type cast (see example script).

Example of PLCSymbolicAddr.SetBool:

Writes the value of a global script variable to a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 12.09.2006
 *
 * Function:
 * Read-in the value of a local script variable
 * and write the value to a symbolic PLC address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mybool      BOOL        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_bool   BOOL
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;
   bool bar = false;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_bool");

   // Read value from global script variable
   foo = Firmware.ScriptVar.GetUInt(Project.ScriptVar.myuint);

   // Set the boolean expression
   if (foo == 1u)
   {
```

```
        bar = true;
    }
    if (foo == 0u)
    {
        bar = false;
    }

    // Write the value to the symbolic address
    Firmware.PLCSymbolicAddr.SetBool(Project.ScriptVar.rcode1,
                                     Project.ScriptVar.symbaddr,
                                     bar,
                                     1u,                          // Station number
                                     0u);                         // Element number
}
```

SetInt(return variable, address script variable, value, station address, element number)

Writes a value of the 0x04 INT32 type with a length of 4 bytes to a symbolic PLC address.

Method	Return type
SetInt	bool
Return value = TRUE	

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	int
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.SetInt:

Writes the value of a global script variable to a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 12.09.2006
 *
 * Function:
 * Read-in the value of a local script variable
 * and write the value to a symbolic PLC address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myint       INT         1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_int    UINT
 */

public void Execute()
{
   // Declaration of local variable
   int foo = 0;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_int");

   // Read value from global script variable
   foo = Firmware.ScriptVar.GetInt(Project.ScriptVar.myint);

   // Write the value to the symbolic address
   Firmware.PLCSymbolicAddr.SetInt(Project.ScriptVar.rcode1,
                                   Project.ScriptVar.symbaddr,
                                   foo,
                                   1u,                          // Station number
                                   0u);                         // Element number
}
```

SetUInt(return variable, address script variable, value, station address, element number)

Writes a value of the 0x07 UINT32 type with a length of 4 bytes to a symbolic PLC address.

Method	Return type
SetUInt	bool
Return value = TRUE	

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.SetUInt:

Writes the value of a global script variable to a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 11.08.2006
 *
 * Function:
 * Read-in the value of a local script variable
 * and write the value to a symbolic PLC address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myuint      UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_uint   UINT
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_uint");

   // Read value from global script variable
   foo = Firmware.ScriptVar.GetUInt(Project.ScriptVar.myuint);

   // Write the value to the symbolic address
   Firmware.PLCSymbolicAddr.SetUInt(Project.ScriptVar.rcode1,
                                    Project.ScriptVar.symbaddr,
                                    foo,
                                    1u,                         // Station number
                                    0u);                        // Element number
}
```

Example of PLCSymbolicAddr.SetUInt for an array item:

Writes the value of a global script variable to an ARRAY item of a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 14.08.2006
 *
 * Function:
 * Read-in the value of an element of an array
 * from a symbolic PLC address into a local script variable
 * and write the value to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myuint      UINT        1
 *  myelement   UINT        1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_array  ARRAY[0..15] OF BYTE
 */

public void Execute()
{
   // Declaration of local variable
   uint foo = 0u;
   uint element = 0u;
   uint bar = 0u;
   uint fcode = 0u;
   uint fsubcode = 0u;

   // Set rcode1 to 0
   Firmware.ScriptVar.SetUInt(Project.ScriptVar.rcode1,0u);

   // Write element number to local script variable
   element = Firmware.ScriptVar.GetUInt(Project.ScriptVar.myelement);

   // Define array element number
   element = 0x80000000 + element;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_array");

   // Write value from global to local script variable
   foo = Firmware.ScriptVar.GetUInt(Project.ScriptVar.myuint);

   // Write the value to the symbolic address
   Firmware.PLCSymbolicAddr.SetUInt(Project.ScriptVar.rcode1,
                                    Project.ScriptVar.symbaddr, // Symbolic address
                                    foo,                        // Value
                                    1u,                         // Station number
                                    element);                   // Element number
}
```

SetDouble(return variable, address script variable, value, station address, element number)

Writes a value into a variable of the 0x83 FLOAT64 type with a length of 8 bytes.

Method	Return type
SetDouble	bool
Return value = TRUE	

Parameter	Type
Return variable (global script variable)	uint
Address script variable (global script variable)	uint
Value	double
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.SetDouble:

Writes the value of a global script variable to a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 13.09.2006
 *
 * Function:
 * Read-in the value of a local script variable
 * and write the value to a symbolic PLC address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mydouble    DOUBLE      1
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_double REAL
 */

public void Execute()
{
   // Declaration of local variable
   double foo = 0d;

   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_double");

   // Read value from global script variable
   foo = Firmware.ScriptVar.GetDouble(Project.ScriptVar.mydouble);

   // Write the value to the symbolic address
   Firmware.PLCSymbolicAddr.SetDouble(Project.ScriptVar.rcode1,
                                      Project.ScriptVar.symbaddr,
                                      foo,
                                      1u,   // Station number
                                      0u);  // Element number
}
```

SetString(return variable, source script variable, address script variable, station address, element number)

Writes a value of the 0x09 C-STRING type from a global script variable to a symbolic PLC address.

Method	Return type
SetString	uint
Return value = length of the written string (number of characters)	

Parameter	Type
Return variable (global script variable)	uint
Source script variable (global script variable)	uint
Address script variable (global script variable)	uint
Station address	uint
Element number	uint

Example of PLCSymbolicAddr.SetString:

Writes a string from a global script variable to a symbolic PLC address.

```
/* Sütron electronic GmbH
 * Version 1, 30.08.2006
 *
 * Function:
 * Writes the value of a global script variable
 * into a symbolic address
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mystring    STRING      20
 *  symbaddr    STRING      20
 *
 * PLC variables (Communication variables list)
 *  NAME        TYPE
 *  .plc_string STRING
 */

public void Execute()
{
   // Write symbolic address to global script variable
   Firmware.ScriptVar.SetString(Project.ScriptVar.symbaddr, ".plc_string");

   // Write the value to the symbolic address
   Firmware.PLCSymbolicAddr.SetString(Project.ScriptVar.rcode1,
                                      Project.ScriptVar.mystring, // Source variable
                                      Project.ScriptVar.symbaddr, // Symbolic address
                                      1u,                         // Station number
                                      0u);                        // Element number
    // Clear global script variable
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring, "");
}
```

2.6.5 Methods for ScriptVar Class

With the methods of this class you access global script variables to write values directly or by means of local script variables to them or read their values and return them to local script variables. Additional methods perform a typecast from INT and DOUBLE to STRING and vice versa.

GetBool(script variable)

GetInt(script variable)

GetUInt(script variable)

GetDouble(script variable)

Reads the value of a script variable from the respective data type.

Method	Return type
GetBool	bool
GetInt	int
GetUInt	uint
GetDouble	double

Parameter	Type
Script variable (global script variable)	uint

Example of ScriptVar.GetBool, ScriptVar.GetInt, ScriptVar.GetUInt and ScriptVar.GetDouble:

Writes the value from global script variables to local script variables of their respective data types.

```
/* Sütron electronic GmbH
 * Version 1, 14.08.2006
 *
 * Function:
 * Writes the values of global script variables to local script variables of the respective
 * data types.
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mybool      BOOL        1
 *  myint       INT         1
 *  myuint      UINT        1
 *  mydouble    DOUBLE      1
 */

public void Execute()
{
    // Declaration of the local variables
    bool myvar1 = false;
    int myvar2 = 0;
    uint myvar3 = 0u;
    double myvar4 = 0d;

    // Write the values to local script variables
    myvar1 = Firmware.ScriptVar.GetBool(Project.ScriptVar.mybool);
    myvar2 = Firmware.ScriptVar.GetInt(Project.ScriptVar.myint);
    myvar3 = Firmware.ScriptVar.GetUInt(Project.ScriptVar.myuint);
    myvar4 = Firmware.ScriptVar.GetDouble(Project.ScriptVar.mydouble);
}
```

SetBool(script variable, value)

Method	Return type
SetBool	bool

Parameter	Type
Script variable (global script variable)	uint
Value	bool

SetInt(script variable, value)

Method	Return type
SetInt	int

Parameter	Type
Script variable (global script variable)	uint
Value	int

SetUInt(script variable, value)

Method	Return type
SetUInt	uint

Parameter	Type
Script variable (global script variable)	uint
Value	uint

SetDouble(script variable, value)

Writes a value to a script variable of the respective data type.

Method	Return type
SetDouble	double

Parameter	Type
Script variable (global script variable)	uint
Value	double

Example of ScriptVar.SetBool, ScriptVar.SetInt, ScriptVar.SetUInt and ScriptVar.SetDouble:

Writes the values of the respective data types to global script variables of the appropriate global script variables. The values are assigned directly or indirectly by means of local script variables.

```
/* Sütron electronic GmbH
 * Version 1, 14.08.2006
 *
 * Function:
 * Writes the values of local script variables
 * to global script variables of the respective
 * data types
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mybool      BOOL        1
 *  myint       INT         1
 *  myuint      UINT        1
 *  mydouble    DOUBLE      1
 */

public void Execute()
{
    // Declaration of the local variables
    bool myvar1 = false;
    int myvar2 = 8;
    uint myvar3 = 123u;
    double myvar4 = 123.456d;

    // Write the values directly
    Firmware.ScriptVar.SetBool(Project.ScriptVar.mybool, false);
    Firmware.ScriptVar.SetInt(Project.ScriptVar.myint, 8);
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.myuint, 123u);
    Firmware.ScriptVar.SetDouble(Project.ScriptVar.mydouble, 123.456d);

    // Write the values indirectly
    Firmware.ScriptVar.SetBool(Project.ScriptVar.mybool, myvar1);
    Firmware.ScriptVar.SetInt(Project.ScriptVar.myint, myvar2);
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.myuint, myvar3);
    Firmware.ScriptVar.SetDouble(Project.ScriptVar.mydouble, myvar4);
}
```

SetString(script variable, string)

Writes a string to a script variable.

Method	Return type
SetString	uint

Parameter	Type
Script variable (global script variable)	string
String	"String"

Example of ScriptVar.SetString:

Writes a string (Hello World) to the global script variable mystring.

```
/* Sütron electronic GmbH
 * Version 1, 13.09.2006
 *
 * Function:
 * Writes a string to a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring    STRING      20
 */

public void Execute()
{
    // Write the string to a global script variable
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring,"Hello World");
}
```

CopyVar(target script variable, source script variable)

Copies the value of a script variable to another script variable.

Both variables must be of the same data type.

Method	Return type
CopyVar	uint

Parameter	Type
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint

Example of ScriptVar.CopyVar:

Writes the value of a globals script variable to another global script variable.

```
/* Sütron electronic GmbH
 * Version 1, 18.09.2006
 *
 * Function:
 * Writes the value from one global script variable to another
 * global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  myvar1      UINT        1
 *  myvar2      UINT        1
 */

public void Execute()
{
    // Copy the value from myvar1 to myvar2
    Firmware.ScriptVar.CopyVar(Project.ScriptVar.myvar2,Project.ScriptVar.myvar1);
}
```

DoubleToString(return variable, double value, fractional digits, target script variable)

Converts the double value to a string. The string is written to the target script variable.

Method	Return type
DoubleToString	void

Parameter	Type
Return variable (global script variable)	uint
Double value	double
Fractional digits	uint
Script variable (target) (global script variable)	uint

Table 2-12 Return variable for method DoubleToString

Error code	Description
0	OK
-1	Double value is larger than destination memory

Example of ScriptVar.DoubleToString:

Performs a typecast from DOUBLE to STRING and writes the string to the global script variable mystring. Enter the double value on a screen and link the input variable (global script variable mydouble) with a post editing script with the content printed below.

```
/* Sütron electronic GmbH
 * Version 1, 19.09.2006
 *
 * Function:
 * Converts the value of a double to a string
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mydouble    DOUBLE      1
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    double doublevar = 0d;

    // Write value from mydouble to doublevar
    doublevar = Firmware.ScriptVar.GetDouble(Project.ScriptVar.mydouble);

    // Converts an double value to a string
    Firmware.ScriptVar.DoubleToString(Project.ScriptVar.rcode1,     // return variable
                                      doublevar,                    // double variable (source)
                                      2u,                           // number of decimals
                                      Project.ScriptVar.mystring);  // string variable (target)
}
```

StringToDouble(return variable, source script variable)

Converts the string contained in a script variable to a double value. The return value contains the double value.

Method	Return type
StringToDouble	double

Parameter	Type
Return variable (global script variable)	uint
Script variable (source) (global script variable)	uint

Table 2-13 Return variable for method StringToDouble

Error code	Description
0	OK
-1	String contains an illegal character

Example of ScriptVar.StringToDouble:

Converts a STRING to DOUBLE and writes the value to the global script variable mydouble. Enter the string on a screen and link the input variable (global script variable mystring) with a post editing script with the content printed below.

```
/* Sütron electronic GmbH
 * Version 1, 13.09.2006
 *
 * Function:
 * Converts a string to the value of an integer
 * and to the value of a double
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mydouble    DOUBLE      1
 *  mystring    STRING      10
 */

public void Execute()
{
    // Declaration of local variable
    double foo = 0d;

    // Converts a string to a double value
    foo = Firmware.ScriptVar.StringToDouble(Project.ScriptVar.rcode1,    // return variable
                                            Project.ScriptVar.mystring); // source (1234.5678)

    // Write the integer value to a global script variable
    Firmware.ScriptVar.SetDouble(Project.ScriptVar.mydouble, foo);
}
```

IntToString(return variable, integer value, target script variable)

Converts an integer value to a string. The string is written to the target script variable.

Method	Return type
IntToString	void

Parameter	Type
Return variable (global script variable)	uint
Integer value	int
Script variable (target) (global script variable)	uint

Table 2-14 Return variable for method IntToString

Error code	Description
0	OK
-1	Integer value is larger than destination memory

Example of ScriptVar.IntToString:

Converts an integer value to a string. The string is written to the global script variable mystring. Enter the integer value on a screen and link the input variable (global script variable myint) with a post editing script with the content printed below.

```
/* Sütron electronic GmbH
 * Version 1, 19.09.2006
 *
 * Function:
 * Converts the value of an integer to a string
 * and the value of a double to a string
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myint       INT         1
 *  mystring    STRING      4
 */

public void Execute()
{
    // Declaration of local variable
    int intvar = 0;

    // Write value from myint to intvar
    intvar = Firmware.ScriptVar.GetInt(Project.ScriptVar.myint);

    // Converts an integer value to a string
    Firmware.ScriptVar.IntToString(Project.ScriptVar.rcode1,    // return variable
                                   intvar,                      // integer variable
                                   Project.ScriptVar.mystring); // target
}
```

StringToInt(return variable, source script variable)

Converts a string contained in a script variable into an integer value. The return variable contains the integer value.

Method	Return type
StringToInt	int

Parameter	Type
Return variable (global script variable)	uint
Script variable (source) (global script variable)	uint

Table 2-15 Return variable for method StringToInt

Error code	Description
0	OK
-1	String contains an illegal character

Example of ScriptVar.StringToInt:

Converts a string to an integer value. The integer is written to the global script variable myint. Enter the string on a screen and link the input variable (global script variable mystring) with a post editing script with the content printed below.

```
/* Sütron electronic GmbH
 * Version 1, 13.09.2006
 *
 * Function:
 * Converts a string to an integer value
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  myint       INT         1
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    int foo = 0;

    // Converts a string to an integer value
    foo = Firmware.ScriptVar.StringToInt(Project.ScriptVar.rcode1,    // return variable
                                         Project.ScriptVar.mystring); // source (1234.5678)

    // Write the integer value to a global script variable
    Firmware.ScriptVar.SetInt(Project.ScriptVar.myint, foo);
}
```

2.6.6 Methods for ScriptVarString Class

With the methods of this class you can perform several operations with string variables, like determining the length of a string or fetching string fragments and appending them to another string.

AppendString(script variable, string)

Appends a string to a script variable.

Method	Return type
AppendString	uint
Return value >0 = length of the string Return value 0xFFFFFFFFu = error	

Parameter	Type
Script variable (global script variable)	string
String	"String"

Example of ScriptVarString.AppendString:

Writes the string "Hello " to a global script variable and appends the string "world" to the end. The result is "Hello world".

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Appends a string to the end of a global variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring    STRING      20
 */

public void Execute()
{
    // Set the string "Hello " to the global variable mystring1
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring, "Hello ");

    // Append the string "world" to the global variable mystring1
    Firmware.ScriptVarString.AppendString(Project.ScriptVar.mystring, "world"); //result is "Hello
world"
}
```

AppendVar(target script variable, source script variable)

Appends the string from the source script variable to the end of the target script variable.

Method	Return type
AppendVar	uint
Return value >0 = length of the string Return value 0xFFFFFFFFu = error	

Parameter	Type
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint

Example of ScriptVarString.AppendVar:

Writes the strings "Hello " and "world" to global script variables and appends them to one global script variable. The result is "Hello world".

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Appends a string to the end of a global variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring1   STRING      20
 *  mystring2   STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint stringlength = 0u;

    // Set the strings "Hello " and "world" to the global variables mystring1 and mystring2
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring1, "Hello ");
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring2, "world");

    // Append mystring2 to mystring1, stringlength contains the length of the result string
    stringlength = Firmware.ScriptVarString.AppendVar(Project.ScriptVar.mystring1,
                                                      Project.ScriptVar.mystring2);
}
```

CopySubstring(target script variable, source script variable, start position)

Reads the remaining string starting from the specified start position of the source script variable and inserts it at the end of the string in the target script variable.

Method	Return type
CopySubstring	uint
Return value >0 = length of the string Return value 0xFFFFFFFFu = error	

Parameter	Type
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint
Start position	uint

Example of ScriptVarString.CopySubstring:

Writes the strings "Hello " and "my world" to global script variables and joins them to one global script variable. In this case only the partial string „world“ is appended so the result is "Hello world".

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Appends a partial string to the end of a string
 * the part goes from a start position to the end of the string

 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring1   STRING      20
 *  mystring2   STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint stringlength = 0u;

    // Set the strings "Hello " and "my world" to the global variables mystring1 and mystring2
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring1, "Hello ");
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring2, "my world");

   // Append "world" from mystring2 to mystring1, stringlength contains the length of the result string
    stringlength = Firmware.ScriptVarString.CopySubstring(Project.ScriptVar.mystring1,
                                                          Project.ScriptVar.mystring2,
                                                          3u); //Start position

    // Write stringlength to global script variable for visualization
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.laenge, stringlength);
}
```

CopySubstringLength(target script variable, source script variable, start position, string length)

Reads the string of the specified string length starting from the specified start position of the source script variable and inserts it at the end of the string in the target script variable.

Method	Return type
CopySubstringLength	uint
Return value >0 = length of the string Return value 0xFFFFFFFFu = error	

Parameter	Type
Script variable (target) (global script variable)	uint
Script variable (source) (global script variable)	uint
Start position	uint
String length	uint

Example of ScriptVarString.CopySubstringLength:

Writes the strings "Hello " and "my world is nice" to a global script variable and joins them to one global script variable. In this case only the partial string "world" is appended so the result is "Hello world".

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Appends a partial string to the end of a string
 * the part has a start position and a defined length

 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring1   STRING      20
 *  mystring2   STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint stringlength = 0u;

    // Set the strings "Hello " and "my world is nice" to the global variables mystring1 and mystring2
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring1, "Hello ");
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring2, "my world is nice");

   // Append "world" from mystring2 to mystring1, stringlength contains the length of the result string
    stringlength = Firmware.ScriptVarString.CopySubstringLength(Project.ScriptVar.mystring1,
                                                                Project.ScriptVar.mystring2,
                                                                3u,     // Start position
                                                                5u);    // length of string part
}
```

Length(script variable)

Determines the length of the string in a script variable.

Method	Return type
Length	uint
Return value >0 = length of the string Return value FFFFFFFF = error	

Parameter	Type
Script variable (global script variable)	uint

Example of ScriptVarString.Length:

Determines the length of a string.

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Determines the length of a string

 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint stringlength = 0u;

    // Determine the string length and write it to local variable
    stringlength = Firmware.ScriptVarString.Length(Project.ScriptVar.mystring);

    // Write stringlength to global script variable for visualization
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.laenge, stringlength);
}
```

CopyString(target script variable, start position, string)

Overwrites the contents of the target script variable with the string as of the given position. The start position has to be counted beginning with zero (0).

Method	Return type
CopyString	uint
Return value >0 = length of the string Return value FFFFFFFF = error	

Parameter	Type
Script variable (target) (global script variable)	uint
Start position	uint
String	string

Example of ScriptVarString.CopyString:

Writes the strings "My world is green" and "red" to global script variables and joins them to one global script variable. In this case "green" is substituted by "red". The remaining characters "en" are automatically deleted so the result is "My world is red".

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Copies a string to a global variable from a start position to the end

 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint stringlength = 0u;

    // Set the string "My world is green" to the global variable mystring
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring, "My world is green");

    // Copy "red" to mystring, stringlength contains the length of the result string
    stringlength = Firmware.ScriptVarString.CopyString(Project.ScriptVar.mystring,
                                                       12u,     //Start position
                                                       "red");  // String

    // Write stringlength to global script variable for visualization
    Firmware.ScriptVar.SetUInt(Project.ScriptVar.laenge, stringlength);
}
```

GetChar(script variable, position)

Reads a character from the specified position of a script variable.

Method	Return type
GetChar	uint
Return value >0 = length of the string Return value FFFFFFFF = error	

Parameter	Type
Script variable (global script variable)	uint
Position	uint

Example of ScriptVarString.GetChar:

Writes the string "Hello" to a global script variable and reads a character from the assigned position. The value of the character is written to the return value.

```
/* Sütron electronic GmbH
 * Version 1, 15.08.2006
 *
 * Function:
 * Reads a single character from a global script variable and returns its value

 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  mystring    STRING      20
 */

public void Execute()
{
    // Declaration of local variable
    uint foo = 0u;

    // Set the string "Hello" to the global variable mystring
    Firmware.ScriptVar.SetString(Project.ScriptVar.mystring, "Hello");

    // Read the character "e", foo contains the value of "e"
    foo = Firmware.ScriptVarString.GetChar(Project.ScriptVar.mystring, 1u);
}
```

2.6.7 Methods for SysVar Class

With the methods of this class you access system variables reading or writing. System variables are only available in the Firmware namespace.

GetUInt(return variable, system variable)

Writes the value of a system variable to a global script variable.

Method	Return type
GetUInt	uint
Return value 0 = OK Return value >0 = error	

Parameter	Type
Return variable (global script variable)	uint
System variable	uint

Example of SysVar.GetUInt:

Writes the value of the system variable **SkriptID** to a local script variable.

```
/* Sütron electronic GmbH
 * Version 1, 16.08.2006
 *
 * Function:
 * Writes the value of a system variable to
 * a local script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 */

public void Execute()
{
    // Declaration of local variable
    uint scriptID = 0u;

    // Write the value from system variable to a local script variable
    scriptID = Firmware.SysVar.GetUInt(Project.ScriptVar.rcode1,Firmware.SysVar.SkriptId);
}
```

SetUInt(return variable, system variable, value)

Passes a value to a system variable.

Method	Return type
SetUInt	bool
Return value 0 = OK Return value >0 = error	

Parameter	Type
Return variable (global script variable)	uint
System variable	uint

Example of SysVar.SetUInt:

Writes the value of a local script variable to the system variable **NewScreen**.

```
/* Sütron electronic GmbH
 * Version 1, 16.08.2006
 *
 * Function:
 * Writes the value of a local script variable to
 * a system variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 */

public void Execute()
{
    // Declaration of local variable
    uint screennumber = 4u;

    // Write the value from a local script variable to a system variable
    Firmware.SysVar.SetUInt(Project.ScriptVar.rcode1,Firmware.SysVar.NewScreen,screennumber);
}
```

GetString(return variable, target script variable, source system variable)

Passes the value of a system variable of the type "String" to a script variable.

Method	Return type
GetString	bool
Return value 0 = OK Return value >0 = error	

Parameter	Type
Return variable (global script variable)	uint
Script variable (target) (global script variable)	uint
System variable (source)	uint

Example of SysVar.GetString:

Writes the string from the system variable **IdentName** to a local script variable.

```
/* Sütron electronic GmbH
 * Version 1, 16.08.2006
 *
 * Function:
 * Writes the string of a system variable to
 * a global script variable
 *
 * Global script variables (Script variables list)
 *  NAME        TYPE        LENGTH
 *  rcode1      UINT        1
 *  mystring    STRING      20
 */

public void Execute()
{
    // Write the string from the system variable to the global script variable
    Firmware.SysVar.GetString(Project.ScriptVar.rcode1,
                              Project.ScriptVar.mystring,
                              Firmware.SysVar.IdentName);
}
```

2.7 Data Types for Scripts

Variables can be used with the following data types:

Table 2-16 Data types for script variables

Type	Range
bool	false to true
int	32-bit integer with a sign from -2147483648 to 2147483647
uint	32-bit integer without a sign from 0 to 4294967295
double	64-bit with a single precision in the IEEE754 format

2.8 Keywords

The script interpreter works with the following keywords:

Table 2-17 Keywords for scripts

Word	Description
bool	Indicator for boolean data type
break	Interrupts the script
double	Indicator for double data type
else	Keyword for IF control structure
false	Value for a boolean expression (0)
if	Keyword for IF control structure
int	Indicator for integer data type
private	Local function within a script
return	Terminates a method
true	Value for a boolean expression (1)
uint	Indicator for unsigned integer data type
void	Return type for a method which does not return a value
while	Keyword for WHILE control structure

2.9 Operators

The script interpreter processes the following operators.

Table 2-18 Script operators

Operator	Description
+	Addition (int, uint, double)
-	Subtraction (int, uint, double)
*	Multiplication (int, uint, double)
/	Division (int, uint, double)
<<	Shift to the left (int, uint - second parameter must be 'int'!)
>>	Shift to the right (int, uint - second parameter must be 'int'!)
~	Bit-by-bit complement (int, uint)
^	Bit-by-bit XOR (int, uint, bool)
&	Bit-by-bit AND (int, uint)
|	Bit-by-bit OR (int, uint)
%	Modulo (remainder after a division) (int, uint)
!	Negation (for "bool" only)
&&	Logical AND (for 'bool' only)
||	Logical OR (for 'bool' only)
=	Assignment (all data types)
(typ)	Cast assignment (converts a data type)
()	Specification of the order of operations in an expression (all data types)

2.10 Comparison Operators

The script interpreter processes the following comparison operators.

Table 2-19 Comparison operators for scripts

Operator	Function
<	less than (int, uint, double)
>	greater than (int, uint, double)
==	is equal to (bool, int, uint, double)
!=	not equal to (bool, int, uint, double)
<=	less than or equal to (int, uint, double)
>=	greater than or equal to (int, uint, double)

The comparison operators ==, <= and >= should not be used with the data type "double" since rounding errors occur for this data type.

2.11 Control Structures

You can use the following control structures for scripts.

Table 2-20 Control structures for scripts

Word	Syntax
if	if (boolean expression) statement OR if (boolean expression) statement else statement
while	while (boolean expression) statement

Example for usage of "if":

```
public void Execute()
{
 f1(1);
}
private void f1(int y)
{
 int z;
 int x=2*y;
 if (x>10)
 {
 z=1;
 }
 else
 {
 z=2;
 }
}
```

If the value of x is greater than 10, the value "1" is written to the variable z. If x is smaller than 10, the value "2" is written.

Example for usage of "while":

```
public void Execute ()
{
 f1(0);
}
private void f1(int y)
{
 int x = 0;
 int z = 0;
 while (x<7)
 {
 z = y + x;
 x++;
 }
}
```

If the value of x is smaller than 7, variable z is offset against the starting value of the transferred value (0) plus the value of variable x. The value of x is incremented for each looping process.In this example, variable z will assume the values 0,1, 2, 3, 4, 5, 6.

2.12 Local Variables

Local script variables are placed on the stack of the script interpreter. These variables are not retentive and can only be used within the method.They are capable of passing the value of a local script variable to a local script method as a parameter. Local script variables can not directly be displayed in screens. The lifetime of local script variables is limited to the execution time of the method.

Example of a local script variable:

```
public void Execute()
{
    bool localVar1 = false;
    int localVar2 = 1;
    uint localVar3 = 2u;
    double localVar4 = 3d;
}
```

2.13 Global Variables

Global script variables are:

– Global script variables
– PLC variables and
– System variables.

You can access these variables using methods. The values of these variables can be displayed and edited in screens.

Enter global script variables into the script variable list.

Example for a global script variable (intVar1):

```
public void Execute()
{
}
private void f1 ()
{
 Firmware.ScriptVar.SetInt(Project.ScriptVar.intVar1, 33);
}
```

The memory for global script variables is limited to the following values:

Table 2-21 Memory for global script variables

Device	Memory
BT03A to BT21A, TP11A to TP21A	32 KB

Global script variables are only retentive if the operating device is equipped with a buffered RAM.

System variables are internally known to the system and are not declared separately.

Example for a system variable (RtcSec):

```
// Write the value of the system variable into the local variable "Seconds"
public void Execute()
{
    uint Seconds;
    Seconds = Firmware.SysVar.GetUInt(Firmware.ScriptVar.MyReturncode, Firmware.SysVar.RtcSec);
}
```

A Index

C

Classes 2-7

E

Example script
- ControllerVar
  - GetString 2-11
  - SetString 2-14
  - SetUInt 2-13
- FileAccess
  - FileClose 2-17
  - FileOpen 2-16
  - FileReadBool 2-18
  - FileReadDouble 2-21
  - FileReadInt 2-19
  - FileReadOneByte 2-24
  - FileReadString 2-22
  - FileReadUInt 2-20
  - FileWriteBool 2-25
  - FileWriteDouble 2-28
  - FileWriteInt 2-26
  - FileWriteOneByte 2-30
  - FileWriteString 2-29
  - FileWriteUInt 2-27
- PLCSymbolicAddr
  - GetDouble 2-41
  - GetInt 2-37, 2-38
  - GetListLength 2-36
  - GetString 2-42
  - GetUInt 2-39
  - GetUInt (ARRAY) 2-40
  - SetBool 2-43
  - SetDouble 2-48
  - SetInt 2-45
  - SetString 2-49
  - SetUInt 2-46
  - SetUInt (ARRAY) 2-47
- ScriptVar
  - CopyVar 2-54
  - DoubleToString 2-55
  - GetBool 2-50
  - GetDouble 2-50
  - GetInt 2-50
  - GetUInt 2-50
  - IntToString 2-57
  - SetBool 2-52
  - SetDouble 2-52
  - SetInt 2-52
  - SetString 2-53
  - SetUInt 2-52
  - StringToDouble 2-56
  - StringToInt 2-58
- ScriptVarString
  - AppendString 2-59
  - AppendVar 2-60
  - CopyString 2-64
  - CopySubstring 2-61
  - CopySubstringLength 2-62
  - GetChar 2-65
  - Length 2-63
- SysVar
  - GetString 2-68
  - GetUInt 2-66
  - SetUInt 2-67

I

Important notes 1-1
Intended use 1-1

M

Methods 2-8
- Class ControllerVar 2-8
- Class FileAccess 2-15
- Class FlashFileSystem 2-31
- Class PLCSymbolicAddr 2-35
- Class ScriptVar 2-50
- Class ScriptVarString 2-59
- Class SysVar 2-66

N

Namespaces 2-7

S

Safety notes 1-1
Script
- Comparison operators 2-70
- Control structures 2-71
- Data types 2-69
- Editor 2-6
- Global variables 2-72
- Keywords 2-69
- Local variables 2-72
- Operators 2-70

Scripting with TSwin .net 2-1
Symbols 1-1

T

Target group 1-1

SÜTRON
visualization | control | communication

Sütron electronic GmbH
Kurze Straße 29
D-70794 Filderstadt
Phone: 0049 711 / 77098-0
Fax: 0049 711 / 77098-60
E-Mail: doku@suetron.de
Internet: www.suetron.com